3.1

Single Commodity Maximum Flow Problem

Katta G. Murty, IOE 612 Lecture slides 3

Simple Transformations

1. To get a model with a single source: If many souce nodes with specified availabilities, can introduce **Supersource** And convert problem into one with a single source.

Original model

Source	Availability
1	a_1 units
2	a_2
$\vdots$	$\vdots$
r	a_r

2. To get a model with a single sink: If many sink nodes with specified requirements, can introduce **Supersink** and convert problem into one with a single sink.

Original model

Sink	Requirement
1	at least b_1 units
2	exactly b_2
$\vdots$	$\vdots$
r	$\ldots a_r$

Another way to handle requirement of *at least* b_1 at a node 1 is to make lower bound and capacity on arc $(1, t)$ equal to $0, b_1$ respectively and look for a flow that saturates this arc.

3. Transforming node capacities into arc capacities: If node i has transit capacity of c_i split node i into a pair of adjacent nodes (*one,*i_1 *say, that acts as receiving end of node i handling incoming flow; the other,* i_2 *say, that acts as shipping end of node i handling outgoing flow*) as below.

4. Combining parallel arcs into one:

Some results

Given $G = (\mathcal{N}, \mathcal{A}, \ell, k, s, t)$ with $|\mathcal{N}| = n, |\mathcal{A}| = m$ problem is to find (f, v) to:

$$\max \quad v$$

$$\text{s. to} \quad f(i,\mathcal{N}) - f(\mathcal{N},i) \;=\; \begin{cases} v & \text{if } i = s \\ 0 & \text{if } i \neq s \text{ or } t \\ -v & \text{if } i = t \end{cases}$$

$$\ell \leq \; f \; \leq k$$

DEFINITION: Net flow in f across a cut $[X, \bar{X}]$: Let f be a feasible flow vector in G and $[X, \bar{X}]$ a cut separating s and t. Then the net flow in f across this cut is defined to be : $f(X, \bar{X}) - f(\bar{X}, X)$.

Results

1. For any feasible flow vector of value v, the net flow across any cut separating s and t is v.

2. Flow value of any feasible flow vector is $\leq$ capacity of any cut separating s and t. So **maximum flow value $\leq$ minimum cut capacity**.

3. If $\hat{f}$ is a feasible flow of value $\hat{v}$, and $[Y, \bar{Y}]$ is a cut separating s and t, and $\hat{v} = k(Y, \bar{Y}) - \ell(\bar{Y}, Y) =$ capacity of cut $[Y, \bar{Y}]$; then $\hat{f} =$ a maximum flow vector, and $[Y, \bar{Y}] =$ a minimum capacity cut.

4. A feasible flow vector in G is of maximum value iff there exists no FAP from s to t WRT it.

Tree groth routine to find $X =$ set of all nodes i with an FAP from s to i WRT f

Step 1: Plant a tree (often called the **Label tree**) with root node at s, i.e., label s with $\emptyset$.

General tree growth step: Look for nodes i, j satisfying:

Either : (i). $(i, j) \in \mathcal{A}$, i labeled, j unlabeled, and $f_{ij} < k_{ij}$

or : (ii). $(j, i) \in \mathcal{A}$, j unlabeled, i labeled, and $f_{ji} > \ell_{ji}$

If such a pair i, j is found, label j with $(i, +)$ in case (i), or $(i, -)$ in case (ii). Then go to the next tree growth step.

If such a pair i, j don't exist, terminate, X is the set of labeled nodes at this stage.

For each $i \in X$, the FAP is the predecessor path of i in the tree written in reverse order.

If sink is in the label tree (i.e., $t \in X$) (event called **Breakthrough**) then there is an FAP from s to t WRT f.

If label tree stops growing without sink getting labeled (i.e., $t \notin X$, this event called **Nonbreakthrough**), there is no FAP from s to t WRT f, then f is a maximum flow.

5. The max flow min cut theorem: If a feasible flow vector exists in G, Maximum flow value = minimum cut capacity.

Duality Interpretation

For simplicity assume $\ell = 0$. Dual of max flow problem is:

$$\begin{aligned} \min z(\pi, u) &= \sum k_{ij} u_{ij} \\ \text{s. to} \quad \pi_j - \pi_i + u_{ij} &\geq 0 \quad \forall (i,j) \in \mathcal{A} \\ \pi_s - \pi_t &= 1 \\ \pi_t &= 0 \\ u_{ij} &\geq 0 \quad \forall (i,j) \in \mathcal{A} \end{aligned}$$

"$\pi_t = 0$" comes from eliminating the redundant flow conservation eq. of node t.

Theorem: Every extreme pt. of the dual system corresponds to a cut separating s and t through the following:

If $(\bar{\pi}, \bar{u})$ is an extreme pt., then all $\bar{\pi}_i, \bar{u}_{ij}$ are $0 - 1$, and if $\bar{X} = \{i : \bar{\pi} = 0\}$, $X = \{i : \bar{\pi} = 1\}$, then $[X, \bar{X}]$ is a cut separating s and t, and:

$$\bar{u}_{ij} = \begin{cases} 1 & \text{iff } (i,j) \in (X, \bar{X}) \\ 0 & \text{otherwise} \end{cases}$$

and vice versa.

So the min cut problem is exactly the problem of finding an extreme point optimum of the dual.

The difference between the Max cut and min cut Problems

Assume $k > 0$. The problem of finding a **maximum capacity cut** is not equivalent to the LP:

$$\text{Max } \sum k_{ij} u_{ij} \quad \text{s. to constraints in the dual}$$

because the maximum objective value in this LP is $+\infty$.

The max capacity cut problem same as problem of finding a max value extreme point sol. in this unbounded LP, an NP-hard problem.

1. Single path Labeling methods for max flow

To find max flow in $G = (\mathcal{N}, \mathcal{A}, \ell, k, s, t)$.

Phase I: Finds a feasible flow first. If $\ell = 0$, $f = 0$ will do. Algorithms for Phase I when $\ell \neq 0$ are based on applying Phase II on a modified network.

Phase II: Find a max flow beginning with an initial feasible flow f^0.

We discuss algorithms for Phase II first.

1.1 Initial Version: Needs an initial feasible flow vector. Tree growth occurs one branch at a time.

Step 1: Rooted tree planting: Let $\bar{f}$ be the current feasible flow vector of value $\bar{v}$. Label s with $\emptyset$.

Step 2: Tree growth: Look for nodes i, j satisfying one of following.

(i) Forward labeling rule i labeled, j unlabeled, $(i, j) \in \mathcal{A}$, and $\bar{f}_{ij} < k_{ij}$.

(i) Reverse labeling rule i labeled, j unlabeled, $(j, i) \in \mathcal{A}$, and $\bar{f}_{ji} > \ell_{ji}$.

If such pair found, label j with $(i, +)$ [$(i, -)$] under rule (i) [(ii)]. If $j = t$ there is a **breakthrough**, go to Step 3. Otherwise repeat Step 2.

If no such pair, **nonbreakthrough**, go to Step 4.

Step 3: Flow augmentation: Let $\mathcal{P}$ be predecessor path of t written in reverse order beginning with s, an FAP. Let ϵ be

residual capacity of $\mathcal{P}$, and new feasible flow vector $\hat{f}_{ij} = (\hat{f}_{ij})$, where:

$$\hat{f}_{ij} = \begin{cases} \bar{f}_{ij} + \epsilon & \text{if } (i,j) \text{ forward } \mathcal{P} \\ \bar{f}_{ij} - \epsilon & \text{if } (i,j) \text{ reverse on } \mathcal{P} \\ \bar{f}_{ij} & \text{if } (i,j) \text{ not on } \mathcal{P} \end{cases}$$

Chop down present tree (i.e., erase labels on all nodes) and go back to Step 1.

Step 4: Termination: Present flow vector is a maximum flow, terminate. X = set of labeled nodes, $\bar{X}$ = complement of X, then $[X, \bar{X}]$ is a min cut.

Analysis when bounds irrational F & F constructed example to show algo. may not work. Let $\alpha = (-1 + \sqrt{5})/2$, $\beta = \Sigma_{r=0}^{\infty} \alpha^r$.

Not all arcs shown. Other arcs are: $(x_i, y_j), (y_i, y_j), (y_i, x_j)$ $\forall i \neq j$. All lower bounds = 0. All capacities β other than 4 shown in fig. Initial feasible flow f^0 of value 1 shown in fig. F & F constructed a sequence of feasible flows $\{f^r\colon\ r = 0, 1, \ldots\}$ s. th. f^{r+1} obtained by augmenting f^r $\forall r$, and values $v^r \uparrow \beta$, while maximum flow value is 4β.

Analysis when bounds, f^0, integral: Each augmentation step increases flow value by > 1. So algo. finds max flow either finitely or thro' an infinite sequence.

Let $u =$ maximum arc capacity. Then capacity of cut $[\{s\}, \mathcal{N} \backslash \{s\}]$ is $\leq nu$, so max flow value $\leq nu$. So no more than nu augmentations.

Work between consecutive augmentations is at most $O(m)$. So overall complexity is $O(nmu)$, a **pseudopolynomial time algorithm**.

1.2 : Scanning version: Each labeled node is **scanned** to label all unlabeled nodes that can be labeled from it. **List** = set of labeled and unscanned nodes.

Step 1: Rooted tree planting: Let $\bar{f}$ be the current feasible flow vector of value $\bar{v}$. Label s with $\emptyset$. List = $\{s\}$.

Step 2.1: Selecting a node from List to scan: If List = $\emptyset$, go to Step 4. Otherwise, delete a node i from list to scan.

Step 2.2: Scanning node i:

(i) Forward labeling Label all j unlabeled s. th., $(i, j) \in \mathcal{A}$, and $\bar{f}_{ij} < k_{ij}$ with $(i, +)$ and put all such j in List.

(i) Reverse labeling rule Label all j unlabeled, s. th. $(j, i) \in \mathcal{A}$, and $\bar{f}_{ji} > \ell_{ji}$ with $(i, -)$ and put all such j in List.

If t now labeled go to Step 3. Otherwise go to Step 2.1.

Steps 3, 4: Same as in initial version.

Sometimes better than initial version, but worst case behavior same as that of initial version.

1.3: Shortest Augmenting Path method

Main idea: At each stage select FAP to be a shortest (in terms of no. of arcs).

Algorithm: Only change needed from scanning version is to maintain List as a **Q** , i.e., select nodes from List to scan using FIFO discipline.

Theorem: For all data, beginning with any feasible flow vector, this method solves max flow problem after $\leq mn/2$ augmentation steps with an overall complexity of $O(nm^2)$. **Strongly polynomial algo.**

2. Multipath Labeling methods for Phase II

In each step these methods augment along all shortest paths simultaneously. Each FAP converted into a chain by reversing orientation of reverse arcs. All these chains put together yield an acyclic network called an **Auxiliary network WRT present flow** f, $AN(f) = (N(f), A(f))$, a partial subnetwork of residual network $G(f)$.

All lower bounds in $AN(f)$ are 0, and upper bounds are residual capacities. Also $A(f) = A^+(f) \cup A^-(f)$ where:

$(i,j) \in A^+(f) \Rightarrow (i,j) \in \mathcal{A}, f_{ij} < k_{ij}$, and κ_{ij} = capacity in $AN(f) = k_{ij} - f_{ij}$.

$(i,j) \in A^-(f) \Rightarrow (j,i) \in \mathcal{A}, f_{ji} > \ell_{ji}$, and κ_{ij} = capacity in $AN(f) = f_{ji} - \ell_{ji}$.

These methods find a **maximal** or **blocking flow** in $AN(f)$ and augment f in G using it.

We denote flow vectors in $AN(f)$ by $g = (g_{ij})$.

General multipath labeling method for maximum flow

Step 1: Initiate the method with a feasible flow vector f^0 in G.

Step 2: Let f be the current feasible flow vector in G. Construct auxiliary network $AN(f)$. Two possible outcomes.

- Construction stops without t joining auxiliary network. Then f is a maximum flow in G. If X = set of all nodes in auxiliary network, and $\bar{X}$ its complement; $[X, \bar{X}]$ is a min cut separating s and t. Terminate.
- t joins the auxiliary network at some stage. Stop construction and go to Step 3.

Step 3: Find a maximal or blocking flow $g = (g_{ij})$ in the auxiliary network. Different multipath methods use different methods for this.

Step 4: Augmentation: Compute new vector $\hat{f} = (\hat{f}_{ij})$ in

G by

$$\hat{f}_{ij} = \begin{cases} f_{ij} & \text{if } (i,j) \notin A(f) \\ f_{ij} + g_{ij} & \text{if } (i,j) \in A^{+}(f) \\ f_{ij} - g_{ij} & \text{if } (i,j) \in A^{-}(f) \end{cases}$$

Value of $\hat{f}$ in G = (Value of f in G) + (Value of g in $AN(f)$.

Go back to Step 2 with $\hat{f}$.

2.1 Dinic's method:

The auxiliary network used is called **Layered Network** $\mathcal{L}(f) = (N(f), A(f))$. Nodes $N(f)$ in it are partitioned into nonempty subsets $\mathcal{N}_0 = \{s\}, \mathcal{N}_1, \ldots, \mathcal{N}_r$ called **Layers**. If f is not a maximum flow in G the last layer will contain t.

Arcs in layered network always go from one layer to next.

Definition: Length of $\mathcal{L}(f) = -1 +$ number of layers.

Layered network construction

Step 1: Initialization: $\mathcal{N}_0 = \{s\}$.

General step to construct next layer: Let $\mathcal{N}_r$ be last layer constructed. For each $i \in \mathcal{N}_r$ do:

for each j s. th. $(i,j) \in \mathcal{A}$ and $f_{ij} < k_{ij}$; put j in $\mathcal{N}_{r+1}$, and (i,j) in $A^+_{r+1}(f)$ with $\kappa_{ij} = k_{ij} - f_{ij}$.

for each j s. th. $(j,i) \in \mathcal{A}$ and $f_{ji} > \ell_{ji}$; put j in $\mathcal{N}_{r+1}$, and (i,j) in $A^-_{r+1}(f)$ with $\kappa_{ij} = f_{ji} - \ell_{ji}$.

- If $\mathcal{N}_{r+1} = \emptyset$, let $X = \cup_{u=0}^{r} \mathcal{N}_u$ and $\bar{X}$ its complement. f is a maximum flow in G, and $[X, \bar{X}]$ is a min cut, terminate Dinics algo.

- If $t \in \mathcal{N}_{r+1}$, let $N(f) = \cup_{u=0}^{r} \mathcal{N}_u$, $A(f) = \cup_{u=1}^{r+1}(A^+_u(f) \cup A^-_u(f))$, $\mathcal{L}(f) = (N(f), A(f))$ stop construction.

- Otherwise go to next step in construction.

Dinic's Procedure for finding Maximal (Blocking) flow in $\mathcal{L} = (N, A, 0, \kappa, s, t)$

Begins with $g^0 = 0$ and augments flows along FACs detected by DFS.

F denotes current set of eligible arcs to include in FAC. Updated by deleting saturated arcs etc.

Step 1: Initialization: $g^0 = 0$ is current flow in $\mathcal{L}$, $F = A$.

Step 2: Initiate DFS: Initiate DFS by labeling s with $\emptyset$ and making it **current node** .

Step 3: DFS: Let $i \in \mathcal{N}_r$ be current node.

- If F has no arc incident out of i, go to Step 6 if $i = s$ or Step 5 if $i \neq s$.
- If F has some arcs incident out of i, select one, say (i, j). Label j with PI i.
  - If $j = t$, predecessor path of t in reverse order is FAC, go to Step 4.

– If $j \neq t$, make j the new current node and repeat this step.

Step 4: Augmentation: Augment flow using FAC, delete all saturated arcs from F, erase labels on all nodes, and go to Step 2.

Step 5: Updating F: No FAC through current node i. Let $p =$ PI of i. Delete all arcs incident into i from F, make p next current node, go back to Step 3.

Step 6: Stop: Present flow vector g is maximal, terminate.

Theorem: In Dinic's method, each successive layered network is strictly longer than the previous.

Theorem: At most $n - 1$ layered networks need to be constructed in Dinic's method.

2.2: Dinic-MKM (Malhotra−Kumar− Maheswari) method

Auxiliary network used is **Referent**, a partial subnetwork of $\mathcal{L}$.

Procedure for finding blocking flow not based on FACs, but uses new operations called **Flow pushing, flow pulling**.

Backward pass routine to find Referent R from $\mathcal{L}$:

Layered network $\mathcal{L}$ may have nodes other than t in last layer $\mathcal{N}_L$. Flows on arcs incident into them will be 0 in every blocking flow. This removes such 0-flow arcs.

- Delete all nodes other than t from last layer $\mathcal{N}_L$ and all arcs incident at such nodes from $\mathcal{L}$.
- Delete all nodes which have no arcs incident out of them, and all arcs incident into those nodes.

  Repeat this step until no further deletions are possible.

Remaining part of layered network called **referent**. It is a union of simple chains from s to t.

F denotes set of eligible arcs of R at current stage, and Y denotes the set of nodes on arcs in F.

Let g be a feasible flow vector in R, and Y, F the current sets. For each $i \in Y$, WRT g, Y, F we define:

$\alpha(i) = \textbf{In-potential of } i = \Sigma_{(j,i)\in F}(\kappa_{ji} - g_{ji})$

$\beta(i) = \textbf{Out-potential of } i = \Sigma_{(i,j)\in F}(\kappa_{ij} - g_{ij})$

$$\rho(i) = \textbf{flow-potential of } i = \begin{cases} \beta(i) & \text{if } i = s \\ \alpha(i) & \text{if } i = t \\ \min\{\alpha(i), \beta(i)\} & \text{if } i \neq s, t \end{cases}$$

MKM procedure for finding blocking flow in R

Step 1: Initialization: Start with $g^0 = 0$ in $R = (N, A, 0, \kappa, s, t)$. $F = A, Y = N$. Compute $\rho(i) \ \ \forall i \in N$ WRT g^0, F, Y. All these initial $\rho(i)$ will be > 0.

Step 2: Reference node selection: Reference potential $= \rho = \min\{\rho(i) : i \in Y\}$, and p a node that attains this min. p called **reference node**.

Step 3.1: Flow Pushing: Push excess flow of ρ out of p (increase flow on arcs of F in forward star of p 1-by-1 saturating them until total increase reaches ρ). *In this process at most one outgoing arc has flow increase but remains unsaturated.*

This has pushed flow from p to nodes in next layer. For each of those nodes, push the excess flow reaching there, out of that node exactly same way. Repeat for nodes in next layer & continue until all excess flow of ρ units reaches t.

Step 3.1: Flow Pulling: Pull excess flow of ρ into p (in-

crease flow on arcs of F in reverse star of p 1-by-1 saturating them until total increase reaches ρ). *In this process at most one incoming arc has flow increase but remains unsaturated.*

This has pulled flow into p from nodes in previous layer. For each of those nodes, pull the excess flow going out of it, into that node exactly same way. Repeat for nodes in previous layer & continue until all excess flow of ρ units is pulled out of s.

After Steps 3.1, 3.2 are completed, we again have a feasible flow vector.

Step 4: Updating $F, Y, \rho(i)$s: 4.1: Delete all saturated arcs from F.

4.2: If all arcs into (or all arcs out of) a node q are deleted from F, delete q from Y, and all arcs incident at q from F. Repeat until no further deletions are possible.

4.3: Update in, out, and flow potentials of nodes in Y WRT present flow and sets Y, F.

- If $\rho(i) = 0$ for some $i \in Y$, go to Step 5 if $\rho(s)$ or $\rho(t)$ is 0; otherwise if $\rho(s), \rho(t)$ are both > 0 delete all nodes i with

$\rho(i) = 0$ from Y and all arcs incident at them from F. Now apply 4.2, after which go to 4.3 again.

- If $\rho(i) > 0 \quad \forall i \in Y$ go to Step 2.

Step 5: Stop: Present flow vector is blocking, stop procedure.

Theorem: The overall complexity of Dinic-MKM algorithm for maximum flow is $O(n^3)$.

3. Preflow-Push methods for Phase II

Newer algorithms. First consider lower bound vector $\ell = 0$.

A **preflow** is a flow vector (not necessarily feasible) $g = (g_{ij})$ satisfying:

- $0 \leq g \leq k$, i.e., bounds hold on all arcs.
- $\forall i \neq s, t$ **net flow into node** $i = e(i) = g(\mathcal{N}, i) - g(i, \mathcal{N}) \geq 0$.

Preflows first introduced by Karzanov.

Active node: Node i said to be an active node in preflow g iff $e(i) > 0$.

These methods work by pushing excess from active nodes to t, or to s if t is not reachable from them. Terminate when no active nodes left, i.e., when flow vector becomes feasible, it will be a maximum flow.

These methods due to Goldberg, Tarzan, use a **node distance vector** $(d(i))$. $d(i)$ is an estimate of the distance in terms of arcs, from node i to t in the residual network $G(g) = (\mathcal{N}, \mathcal{A}(g), 0, k)$. It satisfies following **Validity conditions**:

- $d(i) \geq 0$ and integer $\forall i$.
- $d(t) = 0, d(s) = n$.
- If $(i, j) \in \mathcal{A}(g)$ then $d_i \leq d_j + 1$.

So, if $d(i) < n$ it is a lower bound on actual distance from i to t in $G(g)$. And if $d(i) \geq n$, $d(i) - n$ is a lower bound on actual distance from s to i in $G(g)$.

Arc $(i, j) \in G(g)$ is said to be **admissible arc** WRT d if: i is an active node and $d(j) = d(i) - 1$.

General Preflow push algorithm

Initialization: Define initial g^0 by

$$g_{ij} = \begin{cases} k_{ij} & \text{if } i = s \\ 0 & \text{otherwise} \end{cases}$$

Define initial d^0 by one of the following two choices: Simplest choice is to take $d^0(i) = n$ if $i = s$, 0 otherwise. The better choice is to take $d^0(i)$ to be the distance labels obtained in a backwards BrFS of $G(g^0)$ starting at node t.

General Step: Let (g, d) be present preflow, distance label pair.

If no active nodes, g is a maximum feasible flow, terminate.

If active nodes exist, perform one of following two operations.

Push: Select an active node i and an admissible arc (i, j) incident out of it in $G(g)$.

If (i, j) has a + label [$-$ *label*] in $G(g)$ increase g_{ij} [*decrease* g_{ji}] by $\epsilon = \min\{e(i), \kappa_{ij}\}$.

This push operation is saturating if $\epsilon = \kappa_{ij}$, nonsaturating

otherwise. It decreases $e(i)$ by ϵ, and increases $e(j)$ by ϵ if $j \neq t$.

Relabel: Select an active node i with no admissible arcs incident out of it in $G(g)$. So, j s. th. $(i,j) \in \mathcal{A}(g) \Rightarrow d(i) \leq d(j)$.

Change $d(i)$ to $\min\{1 + d(j) : \quad j \text{ s. th. } (i,j) \in \mathcal{A}(g)\}$.

This relabel operation resets $d(i)$ to highest value allowed by validity conds. It creates at least one admissible arc at active node i on which a push operation can be carried out next.

Complexity of this algo. $\leq O(n^2 m)$, & can be improved to $O(n^3)$ by rules for selecting active nodes, admissible arcs, & order in which push, relabel are applied.

How to find max flow by preflow push when $\ell \neq 0$?

Needs two phases. Phase I (next section) finds a feasible flow vector by applying the max flow algorithm on a modified network in which all lower bounds are 0, so this can be solved by the above preflow push algo.

Phase II : Let $\bar{f}$ be the feasible flow vector obtained in Phase I. Apply the above preflow push algo. to find a max flow in $G(\bar{f})$ $= (\mathcal{N}, \mathcal{A}(\bar{f}), 0, \kappa, s, t)$, let it be $h = (h_{ij})$. Then a maximum flow in G is $\hat{f} = (\hat{f}_{ij})$ where:

$$\hat{f}_{ij} = \begin{cases} \bar{f}_{ij} + h_{ij} & \text{if } (i,j) \in \mathcal{A}(\bar{f}) \text{ with } + \text{ label } \& \ h_{ij} > 0 \\ \bar{f}_{ij} - h_{ji} & \text{if } (j,i) \in \mathcal{A}(\bar{f}) \text{ with } - \text{ label } \& \ h_{ji} > 0 \\ \bar{f}_{ij} & \text{otherwise} \end{cases}$$

Phase I for problems with $\ell \neq 0$

Assume $0 \leq \ell \leq k$. Purpose to find a feasible flow vector or establish infeasibility. In G compute $\ell(i, \mathcal{N}), \ell(\mathcal{N}, i)\ \forall i \in \mathcal{N}$, and $\ell(\mathcal{N}, \mathcal{N})$.

A Feasible circulation is a flow vector satisfying bounds on all arcs, in which flow conservation holds at every node (flow in = flow out at every node).

Modify G by adding an artificial arc (t, s) with $\ell_{ts} = 0, k_{ts} = \infty$, and call modified network G^1.

If f is a feasible flow vector of value v in G, $(f, f_{ts} = v)$ is a feasible circulation in G^1 & vice versa. So, Phase I problem can be solved by finding a feasible circulation in G^1.

In G^1 since we want to find a feasible circulation, there are no source and sink, and s, t are like any other nodes.

Now we transform the problem of finding a feasible circulation in G^1 into a max flow problem on another modified network G^* in which all arc lower bounds are zero. G^* is obtained from G^1

by making following changes:

- Add artificial source, sink nodes s^*, t^* to G^1.
- $\forall i$ s. th. $\ell(i, \mathcal{N}) \neq 0$, add the artificial arc (i, t^*) with lower bound 0, capacity $= \ell(i, \mathcal{N})$.
- $\forall i$ s. th. $\ell(\mathcal{N}, i) \neq 0$, add the artificial arc (s^*, i) with lower bound 0, capacity $= \ell(\mathcal{N}, i)$.
- $\forall (i, j) \in \mathcal{A}$ with $\ell_{ij} \neq 0$, change its lower bound to 0, and capacity to $k_{ij} - \ell_{ij}$. Resulting network is G^*.

Phase I Algorithm: Find maximum flow in G^* from s^* to t^*. Since all arc lower bounds in G^* are 0, this can be found by any algo. discussed earlier beginning with 0-flow in G^*. Let maximum flow be f^* with value v^*.

- If $v^* < \ell(\mathcal{N}, \mathcal{N})$, there exists no feasible circulation in G^1, and hence no feasible flow in G.
- If $v^* = \ell(\mathcal{N}, \mathcal{N})$, let $\hat{f} = (\hat{f}_{ij} = f^*_{ij} + \ell_{ij} : (i, j) \in \mathcal{A})$. Then $(\hat{f}, f^*_{ts})$ is a feasible circulation in G^1, and $\hat{f}$ is a feasible flow vector in G of value f^*_{ts}.

Existence conditions for a feasible circulation

Theorem: Given $G = (\mathcal{N}, \mathcal{A}, \ell, k)$ with $0 \leq \ell \leq k$, a feasible circulation exists in G iff:

$$k(X, \bar{X}) \geq \ell(\bar{X}, X) \quad \forall X \subset \mathcal{N}, \quad \bar{X} = \mathcal{N} \backslash X$$

Critical capacities

Consider the max flow problem in $G = (\mathcal{N}.\mathcal{A}, 0, k, s, t)$. For a particular arc (i, j) define:

$v(k_{ij})$ = maximum flow value in G as a function of capacity k_{ij} as it varies from 0 to ∞, while capacities of all other arcs remain fixed.

Critical capacity of arc $(i, j) = k^*_{ij} = v(\infty) - v(0)$.